

Week 4 and 5

# Dashboarding Overview

---

© IBM Corporation. All rights reserved.

# What you will learn

---



Explore the benefits of using interactive data applications to improve business performance



Identify different web-based dashboarding tools available in Python

# Dashboard

---

- Real-time visuals simplify business moving parts
- Display Key Performance Indicators (KPI) for analysis
- Help businesses by providing the big picture

Best dashboards answer important business questions

# Scenario

---

Review flight performance of US domestic flights

Yearly report items:

- Top 10 airline carriers in 2019
- Number of flights in 2019
- Number of travelers from California

# Type one: Without dashboard

1.

Reporting_Airline	Flights
WN	13972
DL	10241
AA	9853
OO	8515
UA	6430
YX	3416
MQ	3368
B6	3170
OH	2969
AS	2783

2.

Month	Flights
1	6125
2	5578
3	6553
4	6282
5	6441
6	6583
7	6798
8	6753
9	6140
10	6533
11	6280
12	6550

3.

DestState	DistanceGroup	Flights
AK	10	4
AL	8	1
AR	7	1
AZ	2	361
AZ	3	220
AZ	4	9
CA	1	283
CA	2	2256
CA	3	36
CO	3	19
CO	4	401
CO	5	2
CT	11	2
FL	9	51
FL	10	90
FL	11	26
GA	8	114
GA	9	65
HI	10	168
HI	11	116
ID	2	7
ID	3	60
IL	7	169
IL	8	154
IN	8	16
KY	8	13
KY	9	5
LA	7	33
LA	8	14
MA	11	128

# Type two: With dashboard



# Web-based dashboarding

---



*voila*



Streamlit



# Dashboard tools

---

bokeh

Bowtie

ipywidgets

matplotlib



# Recap

---

In this video, you learned that:

- Dashboard simplifies the dynamic aspects of the business.
- Data can be presented by using different types of dashboards.
- There are different types of dashboarding tools.



Search in course

Search



Tushar Raha

Data Visualization with Python > Week 4 > Additional Resources for Dashboards

< Previous Next >

### Creating Dashboards with Plotly

✓ Video: Dashboarding Overview  
4 min

✓ Reading: Additional Resources for Dashboards  
5 min

▶ Video: Introduction to Plotly  
5 min

⌚ Reading: Additional Resources for Plotly  
10 min

⌚ Ungraded App Item: Plotly Basics: Scatter, Line, Bar, Bubble, Histogram, Pie, Sunburst  
1h

⌚ Practice Quiz: Practice Quiz: Creating Dashboards with Plotly  
5 questions

## Additional Resources for Dashboards

For more information about Dashboards, visit the following links:

[Python dashboarding tools](#) ↗

[John Snow's data journalism](#) ↗

✓ Completed

Go to next item



Like



Dislike



Report an issue



Working with Dash

# Introduction to Plotly

---

© IBM Corporation. All rights reserved.

# What you will learn

---



Explore Plotly and its  
two sub-modules



Discover how to use  
Plotly graph objects  
and Plotly express for  
creating charts

# Plotly: An overview

---

- Interactive, open-source plotting library
- Supports over 40 unique chart types
- Includes various types of charts
- Visualizations can be:
  - Displayed in Jupyter notebook
  - Saved to HTML files
  - Used in developing Python-built web applications

# Plotly Sub-modules

---

- Plotly Graph Objects: Low-level interface to figures, traces, and layout

```
plotly.graph_objects.Figure
```

- Plotly Express: High-level wrapper

# Using `plotly.graph_objects`

---

```
# Import required packages
import plotly.graph_objects as go
import plotly.express as px
import numpy as np

# Set random seed for reproducibility
np.random.seed(10)
x = np.arange(12)

# Create random y values
y = np.random.randint(50, 500, size=12)
```

# Using `plotly.graph_objects`

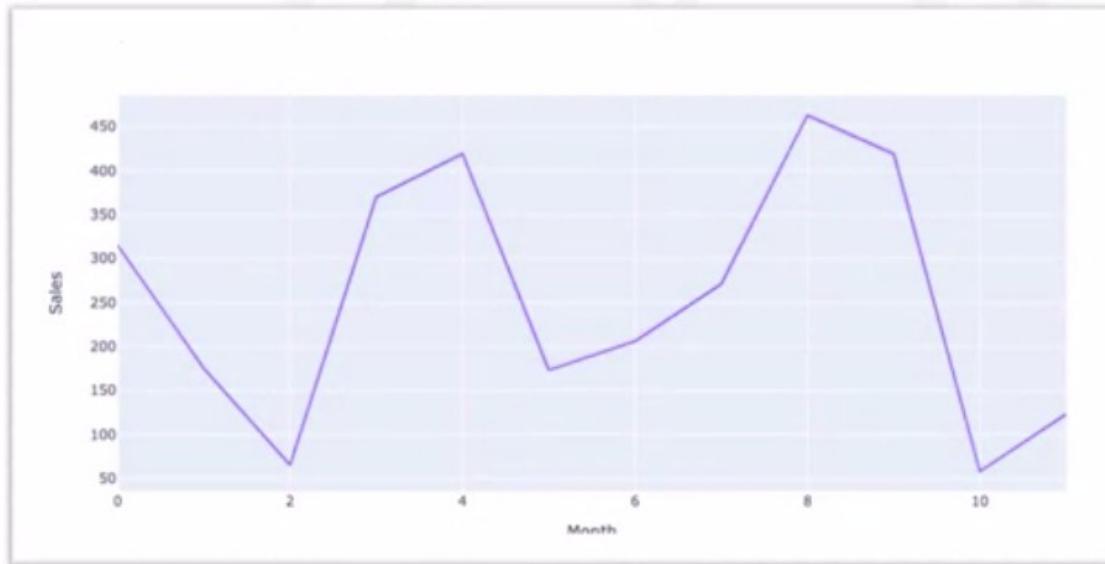
---

- Plotly.graph contains a JSON object
- Here, `go` is the plotly JSON object
- Chart is plotted by updating the values
- Figure is created by adding a trace

```
fig = go.Figure(data=go.Scatter(x=x, y=y))
```

# Using `plotly.graph_objects`

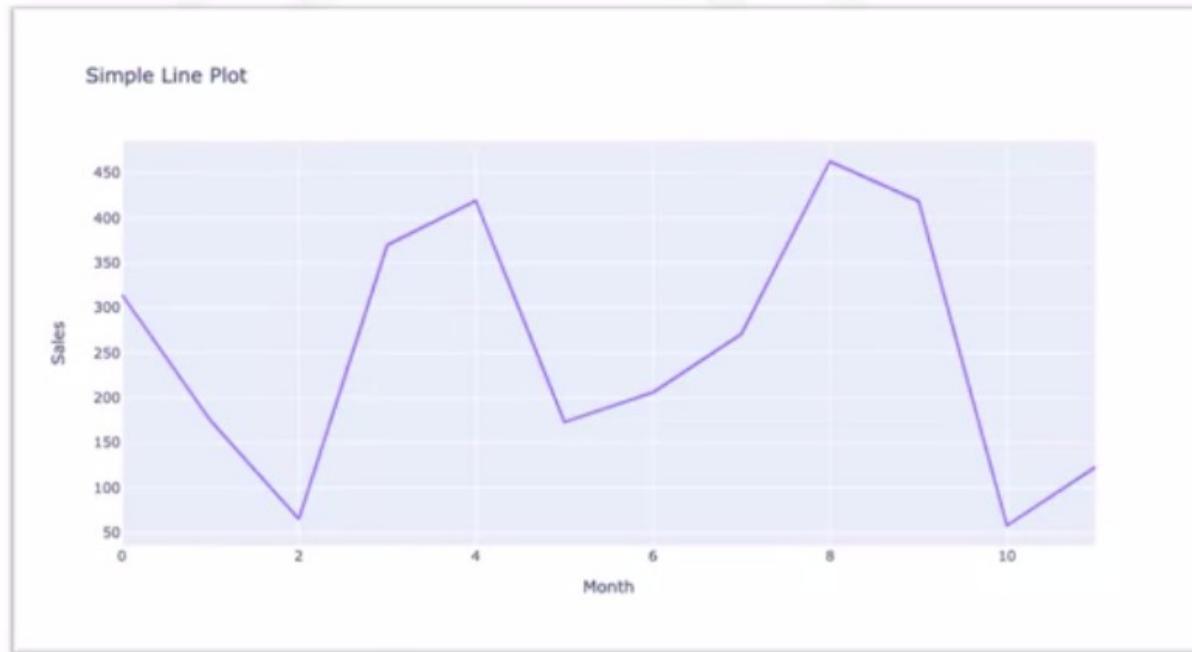
```
fig = go.Figure(data=go.Scatter(x=x, y=y))  
fig.update_layout(title='Simple Line Plot', xaxis_title='Month', yaxis_title='Sales')  
fig.show()
```



# Using plotly.express

```
# Entire line chart can be created in a single command
```

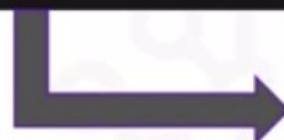
```
fig = px.line(x=x, y=y, title='Simple Line Plot', labels=dict(x='Month', y='Sales'))  
fig.show()
```



# Dataset for Lab

## Data Asset eXchange

Explore useful and relevant data sets for enterprise data science



Dataset | CSV

Airline Reporting Carrier  
On-Time Performance  
Dataset

November 23, 2020



Dataset | CSV

COVID-19 Questions

October 1, 2020



# Dataset for Lab

The screenshot shows the IBM Developer website's "Artificial intelligence" section. The main content is titled "Airline Reporting Carrier On-Time Performance Dataset". It includes a brief description of the dataset, which contains US domestic flight information from 1987 to 2020. Below the title are three buttons: "Try the notebook", "Get this dataset", and "Preview the data & notebooks". The "Get this dataset" button is highlighted with a red box. The "Overview" section provides a detailed description of the dataset's purpose and contents. The "Dataset Metadata" section lists the following fields and their values:

Field	Value
Format	CSV <a href="#">CSV</a>
License	CC0 - Public Domain <a href="#">CC0</a>
Domain	Time Series
Number of Records	214,387,626 entries

# Recap

---

In this video, you learned that:

- Plotly is an interactive, open-source plotting library that supports over 40 unique chart types.
- Plotly graph objects is a low-level interface to figures, traces, and layout.
- Plotly express is a high-level wrapper for Plotly. It uses graph objects internally.



Search in course

Search



Tushar Raha

Data Visualization with Python > Week 4 > Additional Resources for Plotly

< Previous Next >

## Creating Dashboards with Plotly

✓ Video: Dashboarding Overview  
4 min

✓ Reading: Additional Resources for Dashboards  
5 min

✓ Video: Introduction to Plotly  
5 min

✓ Reading: Additional Resources for Plotly  
10 min

ⓘ Ungraded App Item: Plotly Basics: Scatter, Line, Bar, Bubble, Histogram, Pie, Sunburst  
1h

ⓘ Practice Quiz: Practice Quiz: Creating Dashboards with Plotly  
5 questions

Working with Dash

To learn more about using Plotly to create dashboards, explore

[Plotly python](#) ↗

[Plotly.graph objects with example](#) ↗

[Plotly express](#) ↗

[API reference](#) ↗

Here are additional useful resources:

[Plotly cheatsheet](#) ↗

[Plotly community](#) ↗

[Related blogs](#) ↗

[Open-source datasets](#) ↗

✓ Completed

Go to next item



[Back](#) Practice Quiz: Creating Dashboards with Plotly  
Practice Quiz • 10 min • 5 total points

## Congratulations! You passed!

Grade received **100%** To pass 80% or higher

[Go to next item](#)

1. Dashboards can provide real-time visuals.

1 / 1 point

- True  
 False

 **Correct**

Correct! Dashboards can provide real-time visuals. With real-time visuals on the dashboard, understanding business moving parts becomes easy.

2. What is Dash?

1 / 1 point

- A framework tool used for building software.  
 A framework application used for creating statistical graphs.  
 Dash is a Python framework tool used for building Plotly applications.

[Back](#) Practice Quiz: Creating Dashboards with Plotly

Practice Quiz • 10 min • 5 total points

2. What is Dash?

1 / 1 point

- A framework tool used for building software.
- A framework application used for creating statistical graphs.
- Dash is a Python framework tool used for building Plotly applications.
- Dash is a Python framework for building web analytic applications.



Correct

Correct! Dash is a Python framework for building web analytic applications.

3. What is Plotly?

1 / 1 point

- An interactive coding library.
- Plotly is an interactive, open-source plotting library that supports over 40 unique chart types.
- An interactive chart that displays statistical data.
- An interactive open-source code.



Correct

Correct! Plotly is an interactive, open-source plotting library that supports over 40 unique chart types.

[Back](#) Practice Quiz: Creating Dashboards with Plotly

Practice Quiz • 10 min • 5 total points

4. A Callback function is a Python function that is automatically called by Dash whenever an input component's property changes.

1 / 1 point

- True  
 False

 Correct

Correct! A Callback function is a Python function that is automatically called by Dash whenever an input component's property changes. Callback functions are decorated with `@app.callback` decorator.

5. What two functions are required to create a callback?

1 / 1 point

- Input and Library  
 Output and Library  
 Input and Outlet  
 Output and Input

 Correct

Correct! The two functions required to create a callback are Output and Input.



**Skills**  
Network

## Introduction to Dash

# Introduction to Dash

---

© IBM Corporation. All rights reserved.

# What you will learn

---



Explain what Dash is  
and its uses



Describe what Dash  
HTML components  
are

# Dash – An Overview

---

- Open-source User Interface Python library from Plotly
- Easy to build GUI
- Declarative and Reactive
- Rendered in a web browser and can be deployed to servers
- Inherently cross-platform and mobile ready

# Dash Components

---

- Core components

```
import dash_core_components as dcc
```

- HTML Components

```
import dash_html_components as html
```

# Core Components

---

- Higher-level components that are interactive and are generated with JavaScript, HTML, and CSS through the React.js library
- Example: Creating a slider, input area, check items, datepicker, and other components

# HTML Components

---

- Components for every HTML tag
- Keyword arguments describe the HTML attributes like style, className, and id

# Recap

---

In this video, you learned that:

- Dash is an Open-Source User Interface Python library for creating reactive, web-based applications.
- It is easy to build Graphical User Interfaces using Dash as it abstracts all technologies required to make the applications.
- There are two components of Dash: Core and HTML components.
- The `dash_core_components` describe higher-level interactive components generated with JavaScript, HTML, and CSS through the React.js library.
- The `dash_html_components` library has a component for every HTML tag.

Coursera | Online Courses X C Additional Resources for Da +

coursera.org/learn/python-for-data-visualization/supplement/jfnA6/additional-resources-for-dash

Python Tutor co... C Program to Ch... Other bookmarks

Update :

coursera Search in course Search

Tushar Raha

Data Visualization with Python > Week 4 > Additional Resources for Dash < Previous Next >

Creating Dashboards with Plotly

Working with Dash

- Video: Introduction to Dash 3 min
- Ungraded Plugin: Overview of Cloud IDE Lab Environment 15 min
- Ungraded App Item: Dash Basics: HTML and Core Components 40 min
- Reading: Additional Resources for Dash 10 min
- Video: Make Dashboards Interactive 5 min
- Reading: Additional Resources for Interactive Dashboards 10 min
- Ungraded App Item: Add

# Additional Resources for Dash

To learn more about Dash, explore

[Complete dash user guide](#)

[Dash core components](#)

[Dash HTML components](#)

[Dash community forum](#)

[Related blogs](#)

✓ Completed Go to next item

Feedback icon

# Make Dashboards Interactive

---

© IBM Corporation. All rights reserved.

# What you will learn

---



Describe the  
Callback function

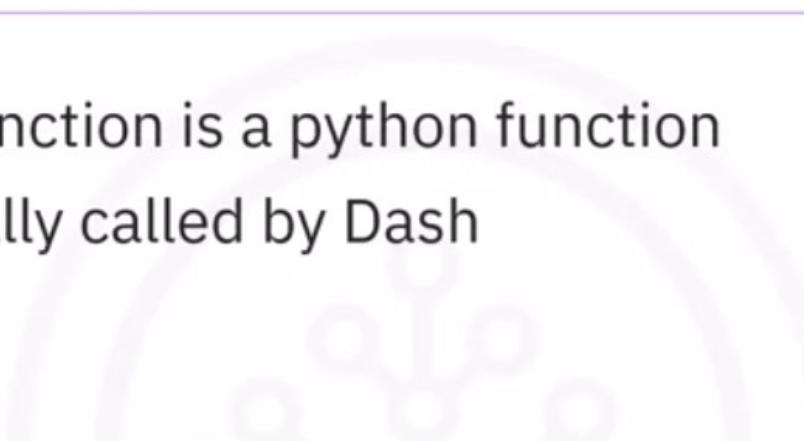


Determine how to  
connect core and  
HTML components  
using Callbacks

# Dash: Callbacks

---

- The Callback function is a python function
- It is automatically called by Dash



Decorator

`@app.callback`

# Dash: Callback function

---

```
def callback_function:  
    ....  
    ....  
    ....  
    return some_result
```

# Dash: Callback function

---

```
def callback_function:  
    ....  
    ....  
    ....  
    return some_result
```

```
@app.callback(Output, Input)
```

# Dash: Callback function

---

Two parameters:

- **Output**

It sets results returned from the callback

- **Input**

It is provided to the callback function

# Callback with one input

---

```
# Import required packages
import pandas as pd
import plotly.express as px
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output

# Read the data
airline_data = pd.read_csv('airline_2m.csv',
                           encoding = "ISO-8859-1",
                           dtype={'Div1Airport': str,
                                  'Div1TailNum': str,
                                  'Div2Airport': str,
                                  'Div2TailNum': str})
```

# Callback with one input

---

```
app = dash.Dash()  
# Design dash app layout  
app.layout = html.Div(children=[ html.H1('Airline Dashboard',  
                                         style={'textAlign': 'center', 'color': colors['text'],  
                                                'font-size': 40}),  
                                         html.Div(["Input: ", dcc.Input(id='input-yr', value='2010',  
                                         type='number', style={'height':'50px', 'font-size': 35}),],  
                                         style={'font-size': 40}),  
                                         html.Br(),  
                                         html.Br(),  
                                         html.Div(dcc.Graph(id='bar-plot')),  
                                         ])
```

# Callback with one input

---

```
@app.callback( Output(component_id='bar-plot', component_property='figure'),
                Input(component_id='input-yr', component_property='value'))

def get_graph(entered_year):
    # Select data
    df = airline_data[airline_data['Year']==int(entered_year)]
    # Top 10 airline carrier in terms of number of flights
    g1 = df.groupby(['Reporting_Airline'])['Flights'].sum().nlargest(10).reset_index()
    # Plot the graph
    fig1 = px.bar(g1, x='Reporting_Airline', y='Flights', title='Top 10 airline carrier in
                                                year ' + str(entered_year) + ' in terms of number of flights')
    fig1.update_layout()
    return fig1

if __name__ == '__main__':
    app.run_server(port=8002, host='127.0.0.1', debug=True)
```

# Callback with one input



# Callback with two inputs

---

```
app = dash.Dash()  
# Design dash app layout  
app.layout = html.Div(children=[ html.H1('Airline Dashboard', style={'textAlign': 'center',  
                           'color': colors['text'], 'font-size': 40}),  
                           html.Div(["Year: ", dcc.Input(id='input-yr', value='2010',  
                                         type='number', style={'height':'50px', 'font-size': 35}),  
                           ], style={'font-size': 40}),  
                           html.Div(["State Abbreviation: ", dcc.Input(id='input-ab',  
                                         value='AL', type='text', style={'height':'50px',  
                                         'font-size': 35})], style={'font-size': 40}),  
                           html.Br(),  
                           html.Br(),  
                           html.Div(dcc.Graph(id='bar-plot')),  
                           ])
```

# Callback with two inputs

---

```
@app.callback( Output(component_id='bar-plot', component_property='figure'),
    [ Input(component_id='input-yr', component_property='value'),
      Input(component_id='input-ab', component_property='value')])

def get_graph(entered_year, entered_state):
    # Select data
    df = airline_data[ (airline_data['Year']==int(entered_year)) &
                      (airline_data['OriginState'] == entered_state)]
    # Top 10 airline carrier in terms of number of flights
    .....
    fig1.update_layout()
    return fig1

if __name__ == '__main__':
    app.run_server(port=8002, host='127.0.0.1', debug=True)
```

# Callback with two inputs



# Callback with two inputs



# Recap

---

In this video, you learned that:

- A callback function is a python function that is automatically called by Dash.
- The `@app.callback` decorator decorates the callback function in order to tell Dash to call it whenever there is a change in the input component value.
- The callback function takes input and output components as parameters and performs operations to return the desired result for the output component.

Components

40 min

**Reading:** Additional Resources for Dash  
10 min **Video:** Make Dashboards Interactive  
5 min **Reading:** Additional Resources for Interactive Dashboards  
10 min **Ungraded App Item:** Add Interactivity: User Input and Callbacks  
30 min **Video:** Understanding the Lab Environment  
7 min **Ungraded App Item:** Flight Delay Time Statistics Dashboard  
1h **Practice Quiz:** Practice Quiz

# Additional Resources for Interactive Dashboards

To learn more about making interactive dashboards in Dash, visit

[Python decorators reference 1](#)

[Python decorators reference 2](#)

[Callbacks with example](#)

[Dash app gallery](#)

[Dash community components](#)

Completed

Go to next item



# Walkthrough the Flight Delay Time Statistics Dashboard Lab

---

© IBM Corporation. All rights reserved.

# What you will learn

---



Describe the Skills  
Network Labs  
Cloud IDE  
environment



Analyze flight on-time data by  
creating line  
graphs with  
dashboard  
components



Create the layout of  
the dash  
application



List the steps to  
run and launch the  
application

# Skills Network Labs (SN Labs) Cloud IDE environment

The screenshot shows a dark-themed interface for a Cloud IDE environment. On the left, there is a sidebar with various icons: a tree (Table of Contents), a document (File), a speech bubble (Edit), a lightbulb (Selection), a megaphone (View), a question mark (Go), a circular arrow (Run), a sun (Terminal), and a person (Help). The main content area has a title "Dash Components" and a logo for "Skills Network". Below the logo, there is a section titled "Objectives" with a list of bullet points. A note says "Estimated time needed: 30 minutes". At the bottom, it mentions "Dataset Used" and "Airline Reporting Carrier On-Time Performance dataset from Data Asset eXchange". The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help.

Table of Contents

File Edit Selection View Go Run Terminal Help

## Dash Components

**Skills Network**

### Objectives

After completing the lab you will be able to:

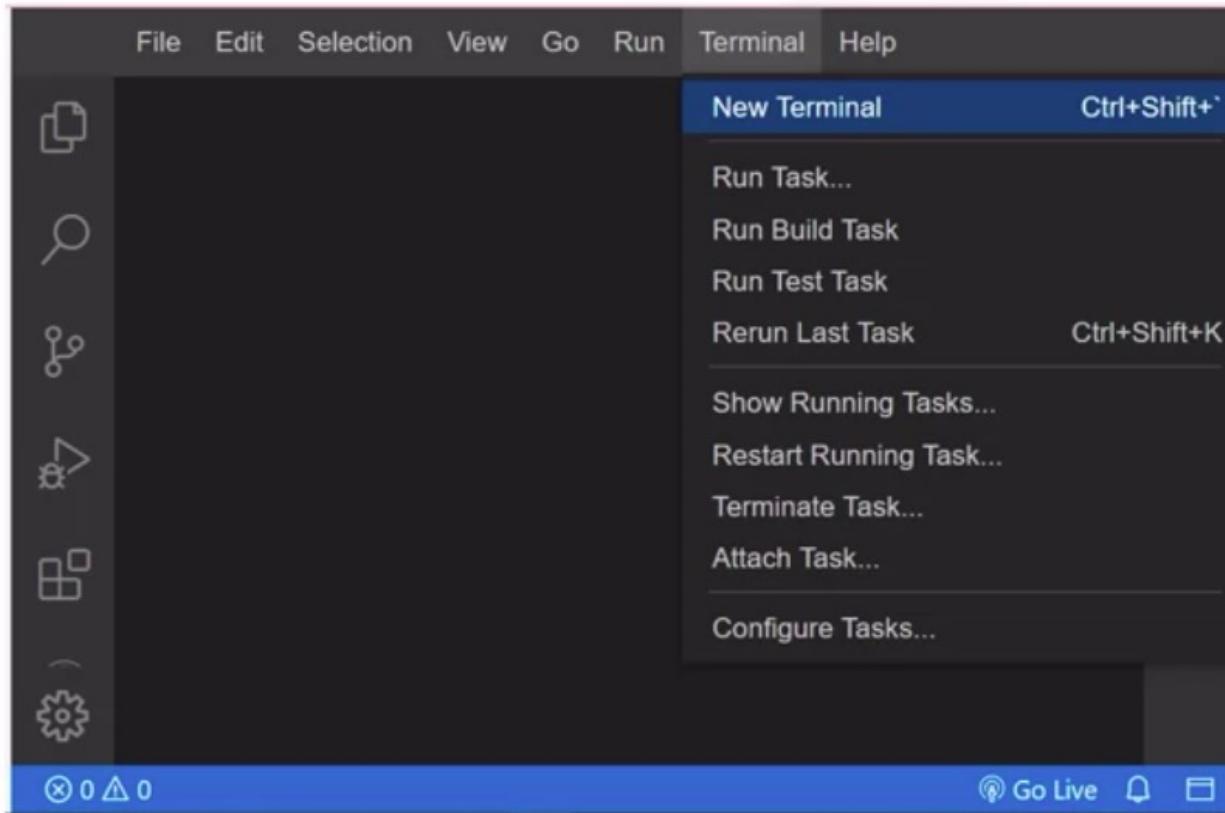
- Know how to add multiple graphs to the dashboard
- Work with Dash Callbacks to handle multiple outputs

Estimated time needed: 30 minutes

### Dataset Used

Airline Reporting Carrier On-Time Performance dataset from Data Asset eXchange

# Open a new terminal



# Install Python packages

---

You will analyze flight delays in a dashboard.



# Dataset

---

1. Airline reporting carrier on-time performance dataset
2. Information on approximately 200 million domestic US flights
3. Basic information about each flight: Time, date, departure and arrival airport, duration of flight delay, and so on

# Install Python packages

---

You will analyze flight delays in a dashboard.

- Monthly average carrier delay
- Monthly average weather delay
- Monthly average national air system delay
- Monthly average security delay
- Monthly average late aircraft delay by reporting airline for the given year.

# Install Python packages

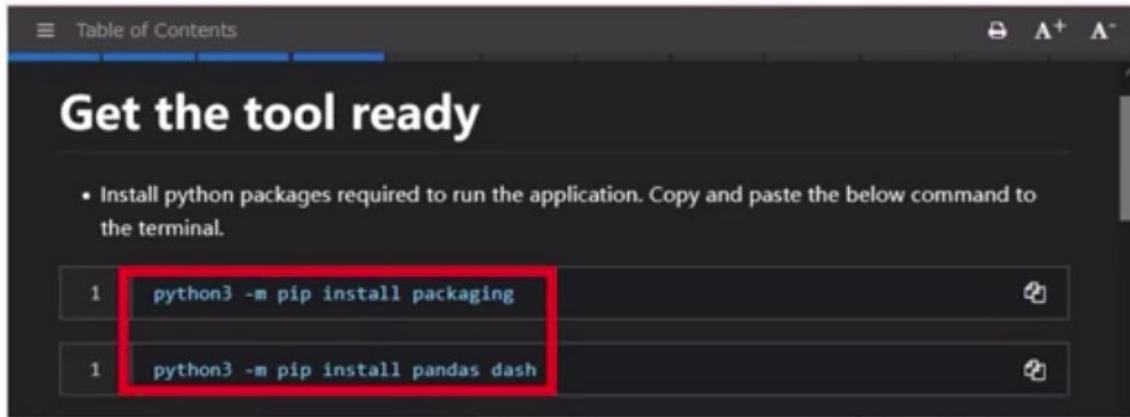


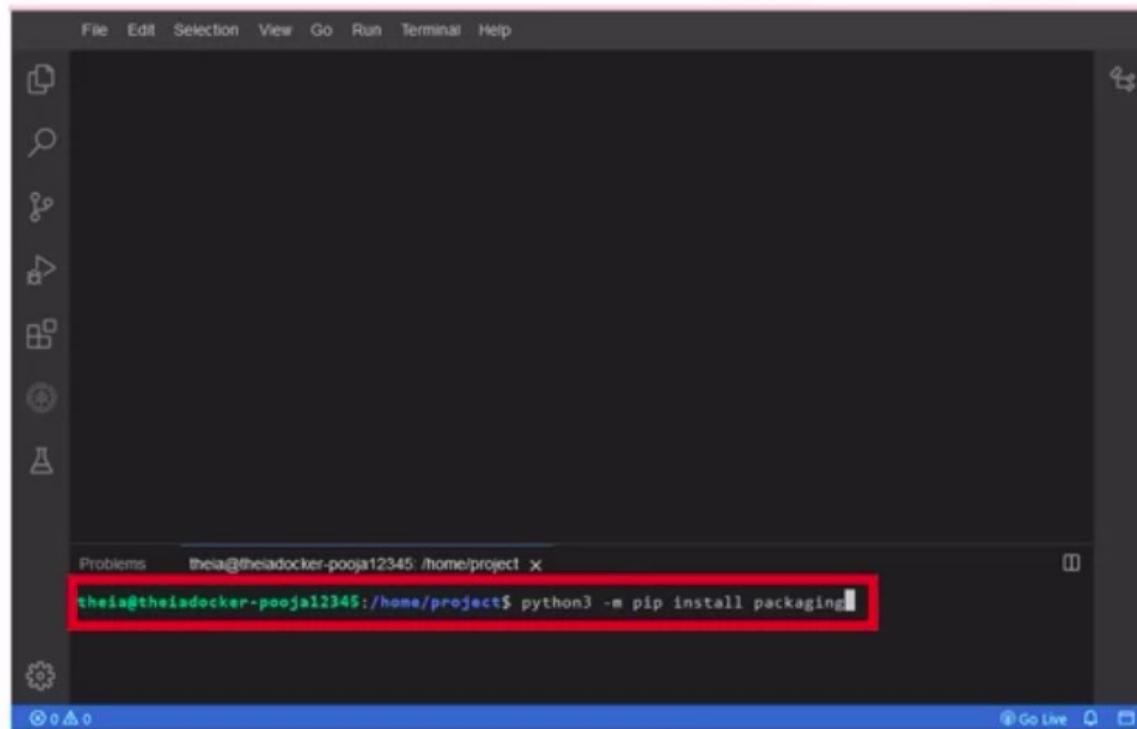
Table of Contents A+ A-

## Get the tool ready

- Install python packages required to run the application. Copy and paste the below command to the terminal.

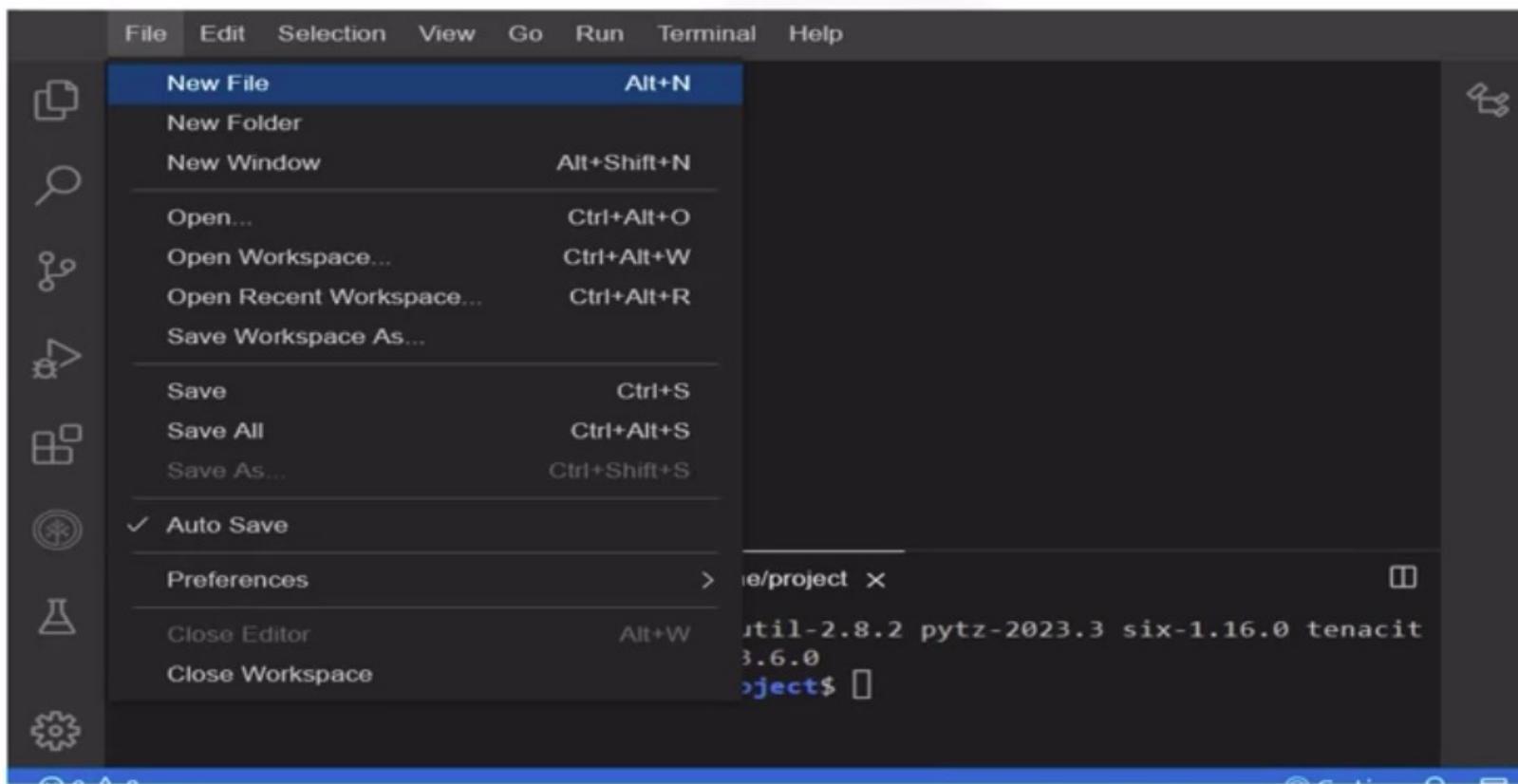
```
1 python3 -m pip install packaging
1 python3 -m pip install pandas dash
```

# Install Python packages

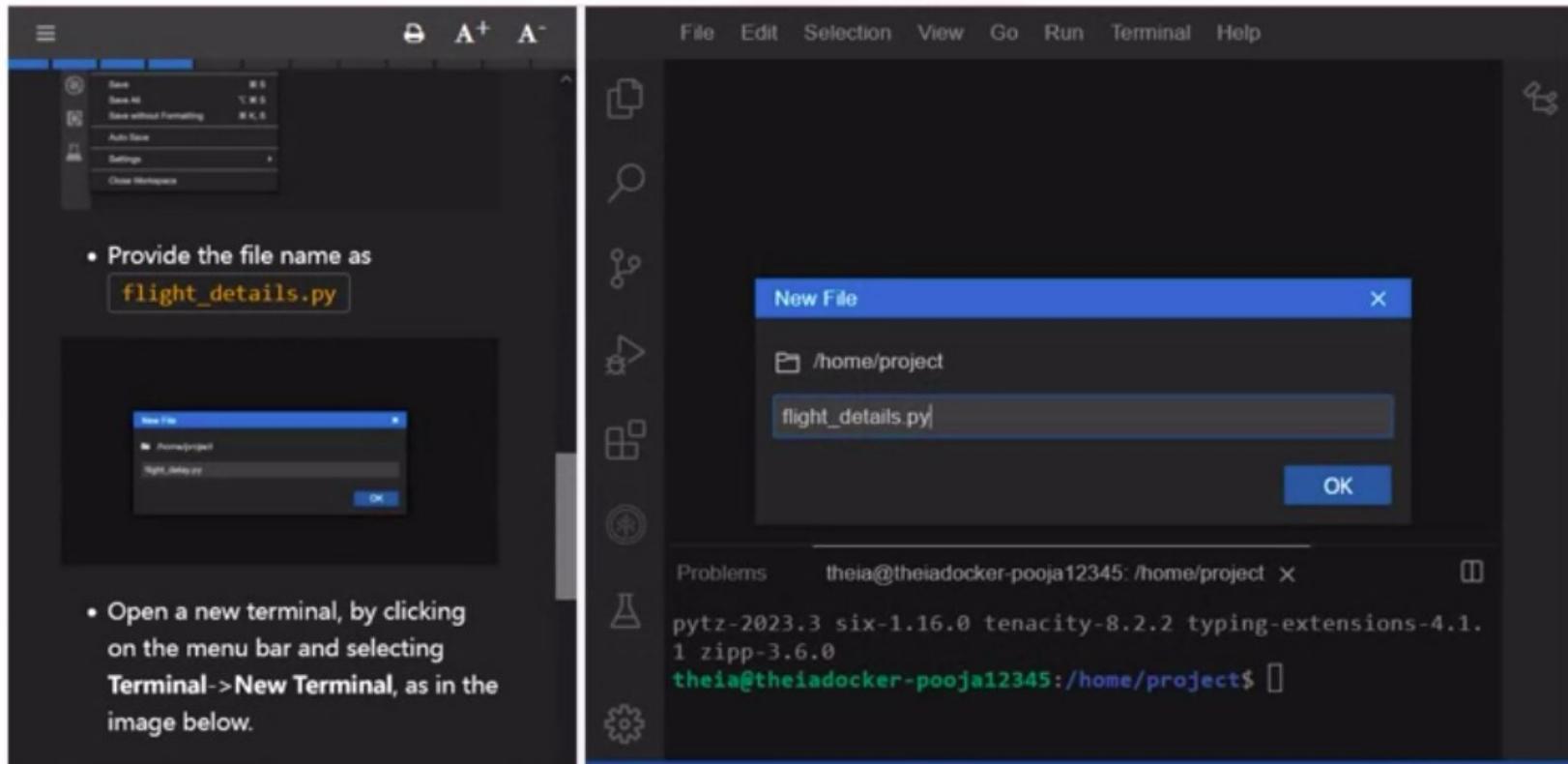


```
File Edit Selection View Go Run Terminal Help  
Problems theia@theiadocker-pooja12345:/home/project x  
theia@theiadocker-pooja12345:/home/projects$ python3 -m pip install packaging
```

# Create a new Python script

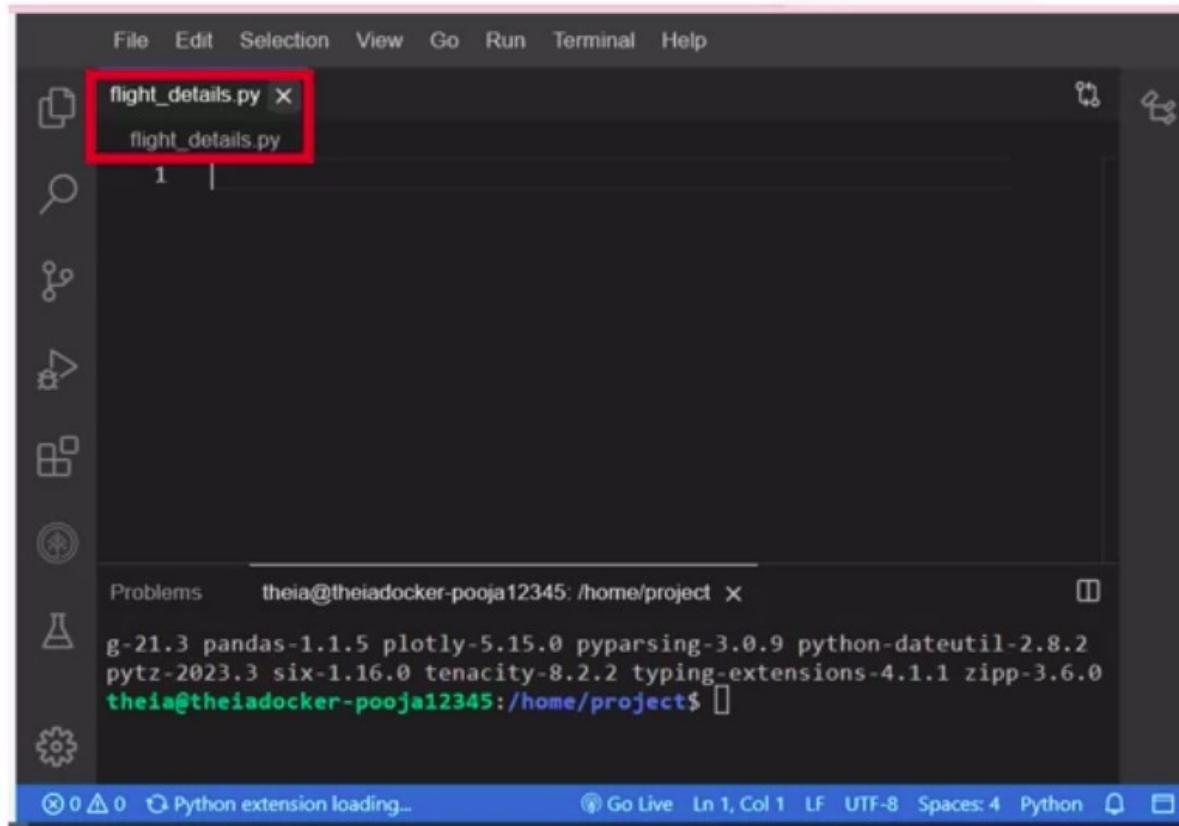


# Create a new Python script



- Provide the file name as  
`flight_details.py`
- Open a new terminal, by clicking on the menu bar and selecting **Terminal->New Terminal**, as in the image below.

# Create a new Python script



The screenshot shows a dark-themed code editor interface. At the top, a menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. Below the menu is a toolbar with various icons: a file icon, a search icon, a refresh icon, a dropdown arrow, a gear icon, and a settings gear icon. A red box highlights the file icon next to the file name 'flight\_details.py'. The main workspace shows the file 'flight\_details.py' with a single line of code: '1 |'. In the bottom left corner, there's a terminal window titled 'Problems' showing the command line output:

```
theia@theiadocker-pooja12345: /home/project
g-21.3 pandas-1.1.5 plotly-5.15.0 pyparsing-3.0.9 python-dateutil-2.8.2
pytz-2023.3 six-1.16.0 tenacity-8.2.2 typing-extensions-4.1.1 zipp-3.6.0
theia@theiadocker-pooja12345:/home/project$
```

At the bottom of the screen, a status bar displays: ⚡ 0 ⚡ 0 Python extension loading.. Go Live Ln 1, Col 1 LF UTF-8 Spaces: 4 Python

# Import libraries and dataset

The screenshot shows a Jupyter Notebook interface with two main panes. The left pane displays a section titled "TASK 1 - Read the data" with instructions and a code snippet. The right pane shows a code editor with a terminal below it.

**Left Pane (Jupyter Notebook):**

- Section Title:** TASK 1 - Read the data
- Text:** Let's start with
- List:**
  - Importing necessary libraries
  - Reading the data
- Text:** Copy the below code to the `flight_delay.py` script and review the code.
- Code Snippet:**

```
1 # Import required libraries
2 import pandas as pd
3 import plotly.graph_objects as go
4 import dash
5 import dash_html_components as html
6 import dash_core_components as dcc
7 from dash.dependencies import Input, Output
8 import plotly.express as px
9
10 # Read the airline data into pandas dataframe
11 airline_data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CA0087ENSkillsNetwork/datasets/airline.csv',
12                             encoding = "ISO-8859-1",
13                             dtype=[('Div1Airport', str, 'Div1TailNum', str),
14                                   ('Div2Airport', str, 'Div2TailNum', str)])
```

**Right Pane (Code Editor and Terminal):**

- Code Editor:** A file named `flight_details.py` is open, showing the first line of code: `1`.
- Terminal:** The terminal shows the user's command prompt: `theia@theiadocker-pooja12345:/home/project$`.
- Bottom Bar:** The bar includes the Python version (`Python 3.6.9 64-bit`), line numbers (`0 ▲ 0`), Go Live, and other system information.

# Import libraries and dataset

The screenshot shows a Jupyter Notebook interface with two code cells. The left cell contains the code for importing libraries and reading the dataset. The right cell shows the code being run in a terminal, with the output showing the command and the user's terminal prompt.

**TASK 1 - Read the data**

Let's start with

- Importing necessary libraries
- Reading the data

Copy the below code to the `flight_delay.py` script and review the code.

```
1 # Import required libraries
2 import pandas as pd
3 import plotly.graph_objects as go
4 import dash
5 import dash_html_components as html
6 import dash_core_components as dcc
7 from dash.dependencies import Input, Output
8 import plotly.express as px
9
10 # Read the airline data into pandas dataframe
11 airline_data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/Intro-to-Python-for-Machine-Learning-and-Data-Science/Flight-Delay-Dataset-Final.csv',
12                             encoding = "ISO-8859-1",
13                             dtype={'Div1Airport': str, 'Div2Airport': str})
```

```
File Edit Selection View Go Run Terminal Help
flight_details.py x
flight_details.py > ...
1 # Import required libraries
2 import pandas as pd
3 import plotly.graph_objects as go
4 import dash
5 import dash_html_components as html
6 import dash_core_components as dcc
7 from dash.dependencies import Input, Output
8 import plotly.express as px
9
10 # Read the airline data into pandas dataframe
11 airline_data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/Intro-to-Python-for-Machine-Learning-and-Data-Science/Flight-Delay-Dataset-Final.csv',
12                             encoding = "ISO-8859-1",
13                             dtype={'Div1Airport': str, 'Div2Airport': str})
14
Problems theia@theiadocker-pooja12345:/home/project x
9 python-dateutil-2.8.2 pytz-2023.3 six-1.16.0 tenacity-
8.2.2 typing-extensions-4.1.1 zipp-3.6.0
theia@theiadocker-pooja12345:/home/project$
```

Python 3.6.9 64-bit ① 0 △ 0 ⚡ Go Live Ln 14, Col 76 LF UTF-8 Spaces: 4 Python

# Create a skeleton for dash application

---

```
# Create a dash application  
app = dash.Dash(__name__)
```

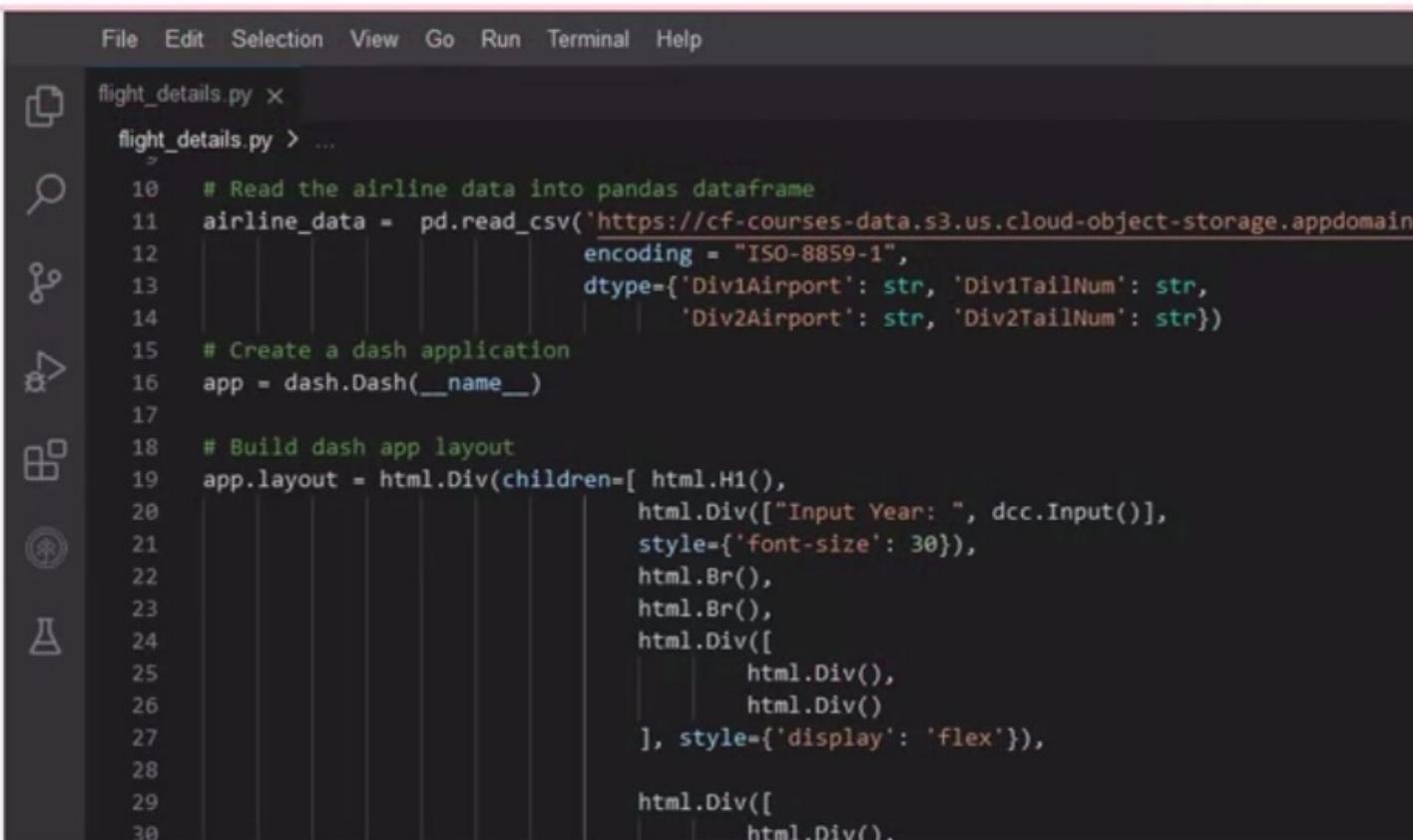
# Create a skeleton for dash application

```
# Build dash app layout
app.layout = html.Div(children=[ html.H1(),
                                  html.Div(["Input Year: ", dcc.Input()],
                                           style={'font-size': 30}),
                                  html.Br(),
                                  html.Br(),
                                  html.Div([
                                              html.Div(),
                                              html.Div()
                                         ], style={'display': 'flex'}),

                                  html.Div([
                                              html.Div(),
                                              html.Div()
                                         ], style={'display': 'flex'}),

                                  html.Div(, style={'width': '65%'})
                               ])
```

# Create a skeleton for dash application



The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there's a sidebar with various icons: a file icon, a search icon, a copy/paste icon, a refresh icon, a file list icon, a cell icon, a terminal icon, and a help icon. The main area displays a Python script named `flight_details.py`. The code reads airline data from a CSV file and creates a Dash application with a simple layout.

```
flight_details.py > ...
10  # Read the airline data into pandas dataframe
11  airline_data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
12  encoding = "ISO-8859-1",
13  dtype={'Div1Airport': str, 'Div1TailNum': str,
14  'Div2Airport': str, 'Div2TailNum': str})
15 # Create a dash application
16 app = dash.Dash(__name__)
17
18 # Build dash app layout
19 app.layout = html.Div(children=[ html.H1(),
20                                 html.Div(["Input Year: ", dcc.Input()],
21                                         style={'font-size': 30}),
22                                 html.Br(),
23                                 html.Br(),
24                                 html.Div([
25                                     html.Div(),
26                                     html.Div()
27                                 ], style={'display': 'flex'}),
28
29                                 html.Div([
30                                     html.Div().
```

# Include title and input

```
html.H1('Flight Delay Time Statistics',  
       style={'textAlign': 'center', 'color':  
'#503D36',  
       'font-size': 30})  
  
html.Div(["Input Year: ", dcc.Input(id='input-year', value='2010',  
                                     type='number', style={'height': '35px', 'font-  
size': 30}),],
```

# Include the graphs

```
html.Div([
    html.Div(dcc.Graph(id='carrier-plot')),
    html.Div(dcc.Graph(id='weather-plot'))
], style={'display': 'flex'})
```

# Prepare data to plot

```
def compute_info(airline_data, entered_year):
    # Select data
    df = airline_data[airline_data['Year']==int(entered_year)]
    # Compute delay averages
    avg_car =
df.groupby(['Month', 'Reporting_Airline'])['CarrierDelay'].mean().reset_index()
    avg_weather =
df.groupby(['Month', 'Reporting_Airline'])['WeatherDelay'].mean().reset_index()
    avg_NAS =
df.groupby(['Month', 'Reporting_Airline'])['NASDelay'].mean().reset_index()
    avg_sec =
df.groupby(['Month', 'Reporting_Airline'])['SecurityDelay'].mean().reset_index()
    avg_late =
df.groupby(['Month', 'Reporting_Airline'])['LateAircraftDelay'].mean().reset_index()
    return avg_car, avg_weather, avg_NAS, avg_sec, avg_late
```

# Callback for real-time dashboard update

```
@app.callback( [  
    Output(component_id='carrier-plot', component_property='figure'),  
  
Input(component_id='input-year', component_property='value')  
  
def get_graph(entered_year):  
  
    # Compute required information for creating graph from the data  
    avg_car, avg_weather, avg_NAS, avg_sec, avg_late =  
compute_info(airline_data, entered_year)  
  
    # Line plot for carrier delay  
    carrier_fig = px.line(avg_car, x='Month', y='CarrierDelay',  
color='Reporting_Airline', title='Average  
  
if __name__ == '__main__':  
    app.run_server()
```

# Launch application

The screenshot shows a terminal window with two panes. The top pane displays the contents of a Python file named `flight_delay.py`. The bottom pane shows the terminal output after running the script.

```
File Edit Selection View Go Run Terminal Help
flight_delay.py > ...
95     # Line plot for nas delay
96     nas_fig = px.line(avg_NAS, x='Month', y='NASDelay', color='Reporting_Airline', title='Average NAS delay ti
97     # Line plot for security delay
98     sec_fig = px.line(avg_sec, x='Month', y='SecurityDelay', color='Reporting_Airline', title='Average securit
99     # Line plot for late aircraft delay
100    late_fig = px.line(avg_late, x='Month', y='LateAircraftDelay', color='Reporting_Airline', title='Average l
101
102    return[carrier_fig, weather_fig, nas_fig, sec_fig, late_fig]
103
104 # Run the app
105 if __name__ == '__main__':
106     app.run_server()

theia@theiadocker-pooja12345:~/home/project >
import dash_html_components as html
flight_delay.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
'import dash_core_components as dcc' with 'from dash import dcc'
import dash_core_components as dcc
Dash is running on http://127.0.0.1:8050/
* Serving Flask app 'flight_delay' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

# Launch application

The screenshot shows a Jupyter Notebook interface with two code cells. The top cell contains Python code for generating four line plots (carrier, weather, NAS, security) and running a Dash app. The bottom cell shows the output of running the app, including a warning about the deprecation of the dash\_core\_components package, the URL where the app is running (http://127.0.0.1:8050), and a note about serving a Flask app.

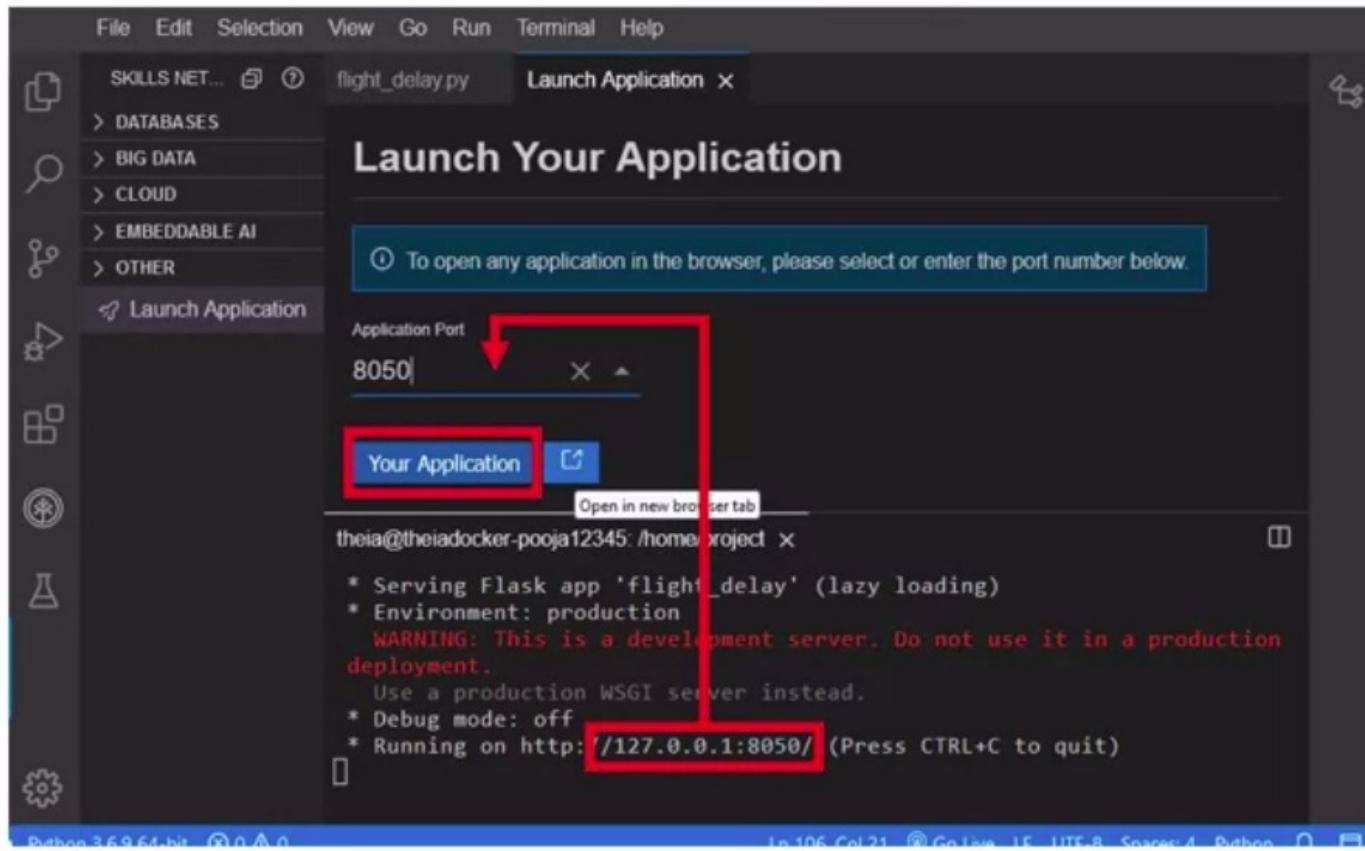
```
flight_delay.py:5: UserWarning:  
The 'dash_core_components' package is deprecated. Please replace  
'import dash_core_components as dcc' with 'from dash import dcc'  
    import dash_core_components as dcc  
Dash is running on http://127.0.0.1:8050/  
  
* Serving Flask app 'flight_delay' (lazy loading)  
* Environment: production  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

# Launch application

The screenshot shows a code editor window with a dark theme. At the top is a menu bar with File, Edit, Selection, View, Go, Run, Terminal, and Help. Below the menu is a toolbar with various icons. The main area displays a Python file named `flight_delay.py`. The code contains several comments and line plots for different types of delays. A red box highlights the `Launch Application` button in the sidebar, which is associated with a tooltip: "Launch your application from a port". Below the code editor, a terminal window shows the command `theia@theiadocker-pooja12345: /home/project` and a message about launching the application.

```
SKILLS NET... ⌂ ⓘ flight_delay.py ×
> DATABASES
> BIG DATA
> CLOUD
> EMBEDDABLE AI
> OTHER
Launch Application
Launch your application from a port
theia@theiadocker-pooja12345: /home/project ×
Use a production WSGI server instead
* Debug mode: off
* Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

# Launch application

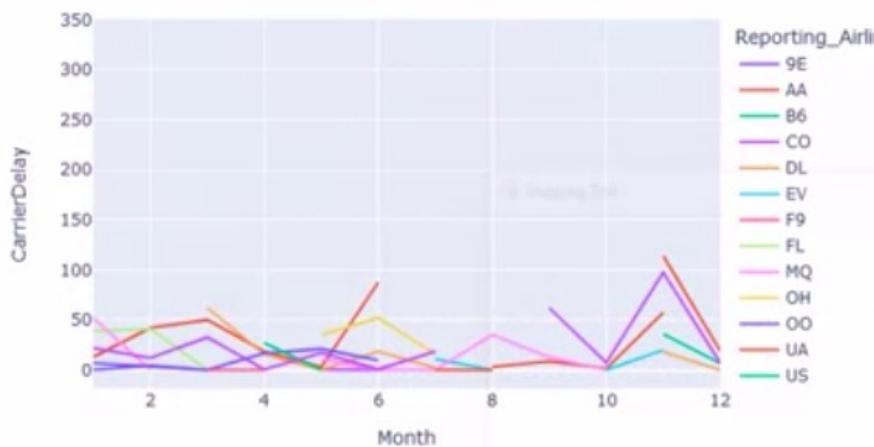


# Launch application

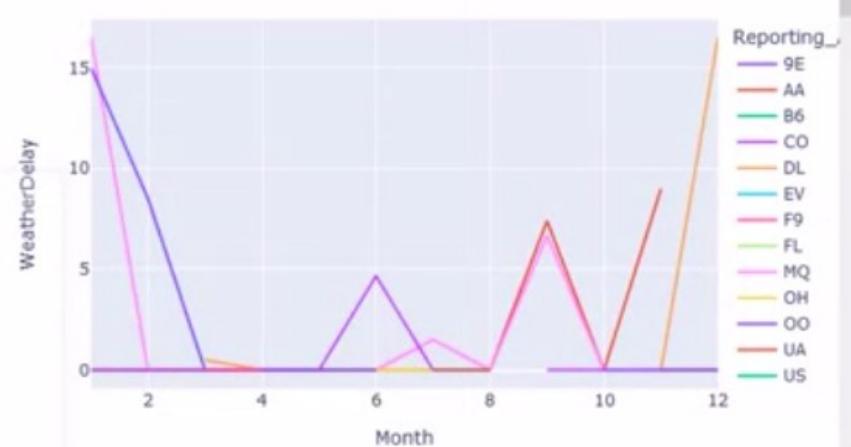
## Flight Delay Time Statistics

Input Year:

Average carrier delay time (minutes) by airline



Average weather delay time (minutes) by airline



# Recap

---

In this video, you learned

- We have walked through the lab where you developed a dashboard with Dash framework on Flight Delay Time Statistics

[Back](#) Practice Quiz: Working with Dash

Practice Quiz • 10 min • 5 total points

## Congratulations! You passed!

Grade received 100% To pass 80% or higher

[Go to next item](#)

1. True or False: Dashboard simplifies the dynamic aspects of the business.

1 / 1 point

True

False

 Correct

Correct! With real-time visuals on the dashboard, understanding business moving parts becomes easy.

2. Fill in the blank. Plotly express is a \_\_\_\_\_ wrapper.

1 / 1 point

High-level

Low-level

[Back](#) Practice Quiz: Working with Dash

Practice Quiz • 10 min • 5 total points

Correct

Correct! With real-time visuals on the dashboard, understanding business moving parts becomes easy.

2. Fill in the blank. Plotly express is a \_\_\_\_\_ wrapper.

1 / 1 point

- High-level
- Low-level

Correct

Correct! Plotly Express is a high-level wrapper for Plotly. It is a recommended starting point for creating the most common figures provided by Plotly because of its simple syntax. It uses graph objects internally.

3. A Callback function is a Python function that is automatically called by \_\_\_\_\_ whenever an input component's property changes.

1 / 1 point

- HTML
- Matplotlib
- Dash

[Back](#) Practice Quiz: Working with Dash

Practice Quiz • 10 min • 5 total points

its simple syntax. It uses graph objects internally.

3. A Callback function is a Python function that is automatically called by \_\_\_\_\_ whenever an input component's property changes.

1 / 1 point

- HTML
- Matplotlib
- Dash
- Plotly

 Correct

Correct! A Callback function is a Python function automatically called by Dash whenever an input component's property changes. Callback functions use the `@app.callback` decorator.

4. Which of the following is the correct way to add the callback decorator?

1 / 1 point

- @app.callback[Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value')]
- @app.callback(Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value'))

[Back](#) Practice Quiz: Working with Dash

Practice Quiz • 10 min • 5 total points

Correct

Correct! A Callback function is a Python function automatically called by Dash whenever an input component's property changes. Callback functions use the `@app.callback` decorator.

4. Which of the following is the correct way to add the callback decorator?

1 / 1 point

- @app.callback[Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value')]
- @app.callback( Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value'))
- @app.callback( Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value'))
- @app.callback{ Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value')}

Correct

Correct! The correct way of adding a callback decorator is the following: @app.callback( Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value')).

5. The callback function takes input and output components as \_\_\_\_\_.

1 / 1 point

Parameters

[Back](#) Practice Quiz: Working with Dash

Practice Quiz • 10 min • 5 total points

- @app.callback( Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value'))
- @app.callback{ Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value')}

 **Correct**

Correct! The correct way of adding a callback decorator is the following: @app.callback( Output(component\_id='bar-plot', component\_property='figure'), Input(component\_id='input-yr', component\_property='value')).

5. The callback function takes input and output components as \_\_\_\_\_.

1 / 1 point

- Parameters
- Matplotlib
- Data
- Perimeter

 **Correct**

Correct! The callback function takes input and output components as parameters and performs operations to return the desired result for the output component.



Search in course

Search



Tushar Raha

## Creating Dashboards with Plotly

### Working with Dash

Video: Introduction to Dash  
3 min

Ungraded Plugin: Overview of Cloud IDE Lab Environment  
15 min

Ungraded App Item: Dash Basics: HTML and Core Components  
40 min

Reading: Additional Resources for Dash  
10 min

Video: Make Dashboards Interactive  
5 min

Reading: Additional Resources for Interactive Dashboards  
10 min

Ungraded App Item: Add

Congratulations! You have completed this module. At this point in the course, you know:

- Dash is an Open-Source User Interface Python library for creating reactive, web-based applications.
- It is easy to build Graphical User Interfaces using Dash as it abstracts all technologies required to make the applications.
- There are two components of Dash: Core and HTML components.
- The dash\_core\_components describe higher-level interactive components generated with JavaScript, HTML, and CSS through the React.js library.
- The dash\_html\_components library has a component for every HTML tag.
- A callback function is a python function that is automatically called by Dash whenever an input component's property changes.
- The @app.callback decorator decorates the callback function in order to tell Dash to call it whenever there is a change in the input component value.
- The callback function takes input and output components as parameters and performs operations to return the desired result for the output component.



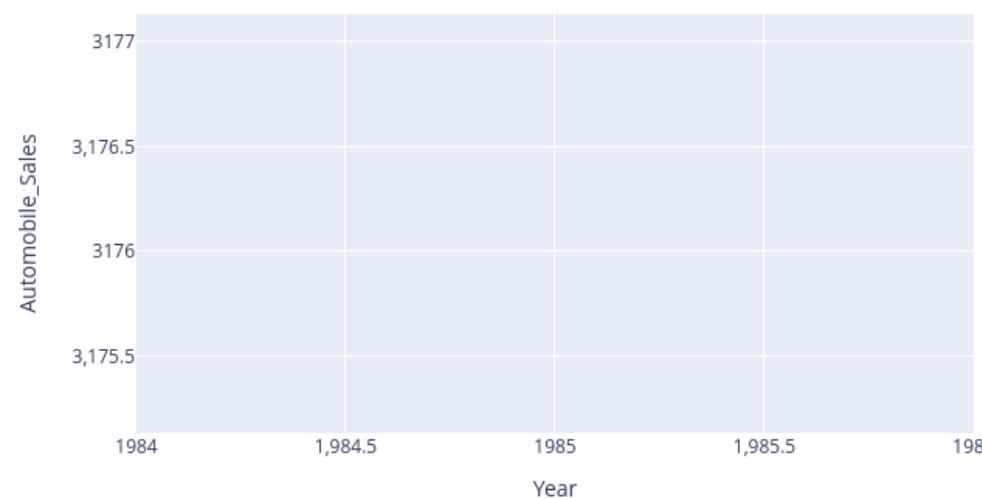
## Automobile Sales Statistics Dashboard

Select Statistics:

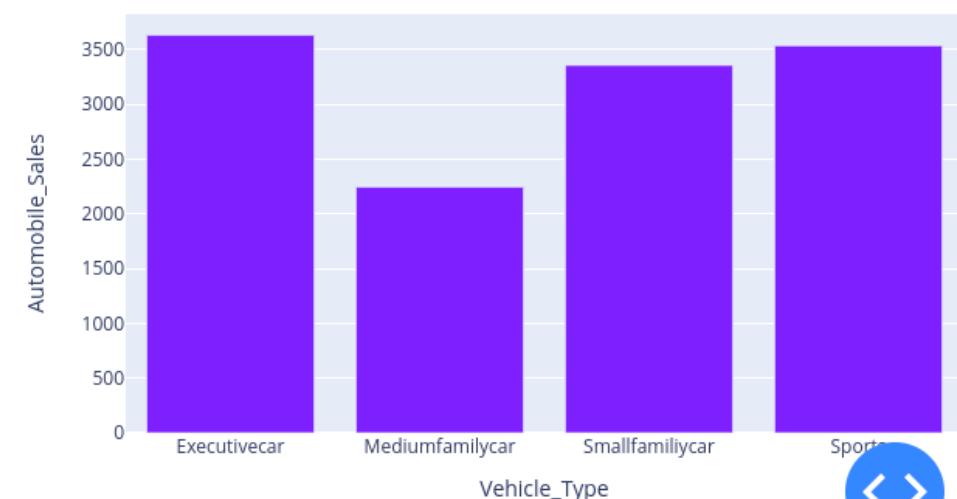
Yearly Statistics

1985

Yearly Automobile Sales



Average Vehicles Sold by Vehicle Type in the year 1985



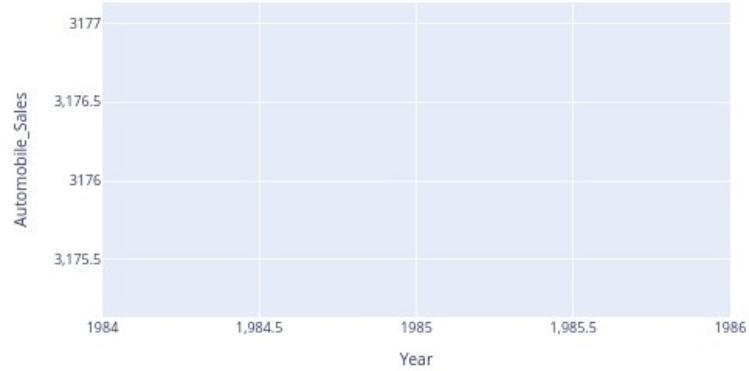
# Automobile Sales Statistics Dashboard

Select Statistics:

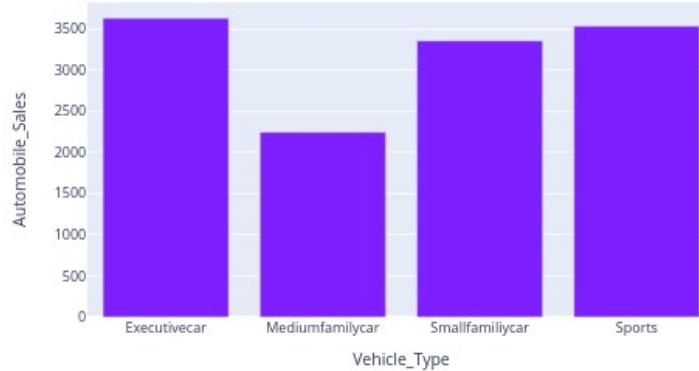
Yearly Statistics

1985

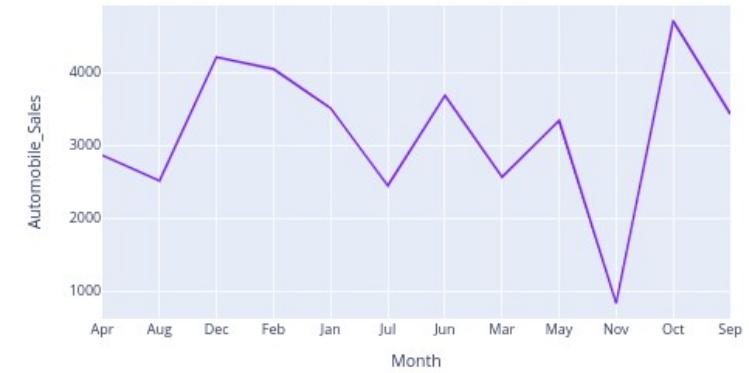
Yearly Automobile Sales



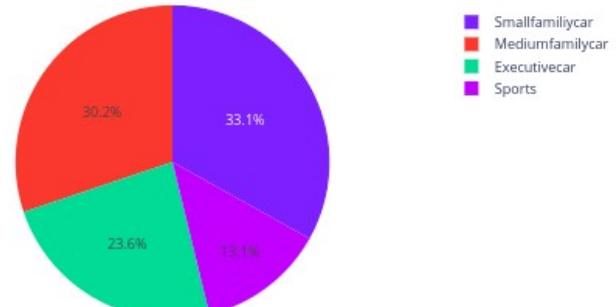
Average Vehicles Sold by Vehicle Type in the year 1985



Total Monthly Automobile Sales



Total Advertisement Expenditure by Vehicle Type in the year 1985



[Back](#)

## Graded Quiz: Creating Dashboards with Plotly and Dash

Graded Quiz • 30 min

Due Oct 22, 11:59 PM IST

## Congratulations! You passed!

Grade received **100%** Latest Submission Grade 100% To pass 70% or higher

Retake the assignment in **7h 56m**

Go to next item

1. With Plotly, where can created web-based visualizations be displayed?

1 / 1 point

- Python
- Jupyter notebook
- Dash
- Matplotlib

Correct

Correct! Web-based visualizations created using Plotly Python can be displayed in Jupyter Notebook, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash.

2. True or False. The Plotly graph objects module provides an automatically generated hierarchy of classes. It is the low-level interface to figures, traces, and layouts.

1 / 1 point

Back

## Graded Quiz: Creating Dashboards with Plotly and Dash

Due Oct 22, 11:59 PM IST

Graded Quiz • 30 min



Correct

Correct! Web-based visualizations created using Plotly Python can be displayed in Jupyter Notebook, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash.

2. True or False. The Plotly graph objects module provides an automatically generated hierarchy of classes. It is the low-level interface to figures, traces, and layouts.

1 / 1 point

 False True

Correct

Correct! The Plotly graph objects module provides an automatically generated hierarchy of classes. It is the low-level interface to figures, traces, and layouts.

3. Fill in the blank. A Plotly.graph contains a \_\_\_\_\_ object which has a dictionary structure.

1 / 1 point

 HTML Matplotlib JSON

[Back](#)

## Graded Quiz: Creating Dashboards with Plotly and Dash

Due Oct 22, 11:59 PM IST

Graded Quiz • 30 min

3. Fill in the blank. A Plotly.graph contains a \_\_\_\_\_ object which has a dictionary structure.

1 / 1 point

- HTML
- Matplotlib
- JSON
- Python

Correct

Correct! A Plotly.graph contains a JSON object which has a dictionary structure.

4. True or False. Real-time visuals on the dashboard simplify the comprehension of various aspects of the business.

1 / 1 point

- False
- True

Correct

Correct! Real-time visuals on the dashboard simplify the comprehension of various aspects of the business. Also, getting the big picture in one place can help



## Graded Quiz: Creating Dashboards with Plotly and Dash

Graded Quiz • 30 min

Due Oct 22, 11:59 PM IST

4. True or False. Real-time visuals on the dashboard simplify the comprehension of various aspects of the business.

1 / 1 point

- False
- True



Correct

Correct! Real-time visuals on the dashboard simplify the comprehension of various aspects of the business. Also, getting the big picture in one place can help businesses make informed decisions, thereby improving performance.

5. Fill in the blank. Matplotlib is a comprehensive library for creating static, animated, and interactive \_\_\_\_\_ in Python.

1 / 1 point

- Graphs
- Visualizations
- Videos
- Charts



Correct

Correct! Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

[Back](#)

## Graded Quiz: Creating Dashboards with Plotly and Dash

Graded Quiz • 30 min

Due Oct 22, 11:59 PM IST

6. True or False. Data can be presented by using different types of dashboards.

1 / 1 point

- False  
 True

Correct

Correct! Data can be presented by using different types of dashboards.

7. True or False. Plotly Express is a high-level wrapper for Plotly.

1 / 1 point

- True  
 False

Correct

Correct! Plotly Express is a high-level wrapper for Plotly. It is a recommended starting point for creating the most common figures provided by Plotly because of its simple syntax. It uses graph objects internally.

Back Graded Quiz: Creating Dashboards with Plotly and Dash  
Graded Quiz • 30 min

Due Oct 22, 11:59 PM IST

8. What is a callback function?

1/1 point

- A decorator
- A line of Code
- A file
- An extension



Correct

Correct! Whenever there is a change in the input component value, the Callback function wrapped by the decorator is called, followed by the update to the output component children in the application layout.

9. True or False. Dash is an Open-Source User Interface Python library for creating reactive, web-based applications.

1/1 point

- True
- False



Correct

Correct! Dash is an Open-Source User Interface Python library for creating reactive, web-based applications. It is enterprise-ready and a first-class member of Plotly's open-source tools.

[Back](#)

## Graded Quiz: Creating Dashboards with Plotly and Dash

Graded Quiz • 30 min

Due Oct 22, 11:59 PM IST



Correct

Correct! Dash is an Open-Source User Interface Python library for creating reactive, web-based applications. It is enterprise-ready and a first-class member of Plotly's open-source tools.

10. Which **two** parameters are needed to decorate a callback function? **Select two.**

1 / 1 point

 Output

Correct

Correct! Output sets results returned from the callback function to a component id. The set input is provided to the callback function to a component id.

 Dash core Input

Correct

Correct! Output sets results returned from the callback function to a component id. The set input is provided to the callback function to a component id.

 Dash



Search in course

Search



Tushar Raha

**Practice Project****Final Project****Course Wrap Up**

Reading: Course Summary  
5 min

Reading: Congratulations and  
Next Steps  
5 min

Reading: Thanks from the Course  
Team  
5 min

**Digital Badge**

# Course Summary

In this course, you learned how to use Data Visualization with Python to create charts and graphs for business or educational purposes. Graphs and charts are essential for communicating statistical information directly and efficiently. In this course, you also learned the history of Data Visualization with Python, why it was created, and who created it. Understanding the creation process of Data Visualization applications and programs is essential for understanding its uses, abilities, and importance. You can apply these skills to other graphing and statistical visualization applications that utilize Python and code.

## Next Steps

After completing this course, you could take similar courses related to Data Visualization with Python, “Python for Data Science AI & Development, Python for Data Science, and Databases and SQL for Data Science with Python courses.”

✓ Completed

Go to next item

